Comment visualiser la grande image AT.S avec Python

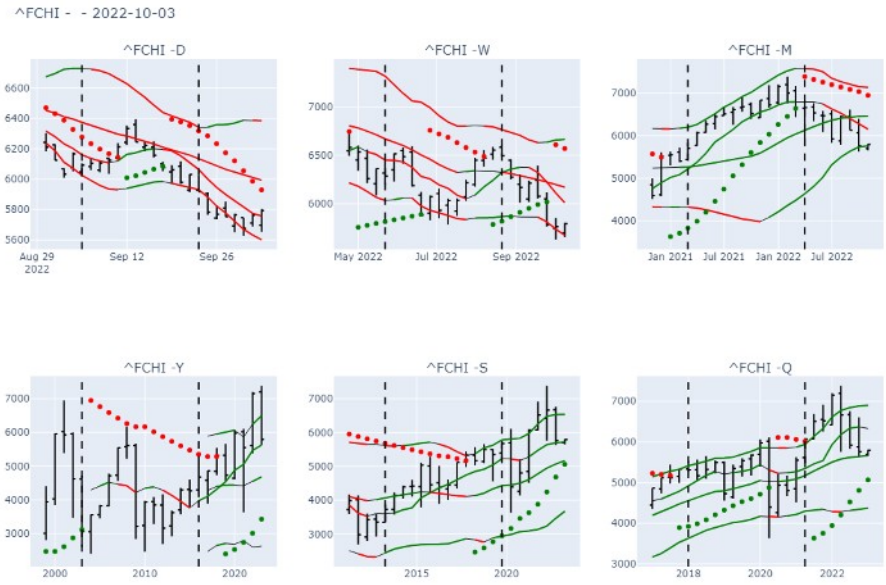
INTRODUCTION

Le langage Python est très populaire pour traiter des données, et grandement utilisé dans le monde de la finance, surtout depuis le développement de la librairie Pandas. Cette librairie permet de manipuler des tables de données très facilement.
Ainsi, Python peut être utilisé pour faire de l'analyse quantitative ou de l'analyse technique sur des actifs financiers. De nombreux codes et tutoriels pour la finance sont disponibles sur internet.

Le système d'analyse technique AT.S utilise des indicateurs relativement simples (moyenne mobile, Bandes de Bollinger, SAR). La principale difficulté technique réside en la présence des 6 unités de temps (Jour, Semaine, Mois, Trimestre, Semestre, Année) pour visualiser la "grande image".

Pourquoi développer le système sous Python quand il est utilisable sous ProRealTime (gratuit) et TradingView (premium) ?
Au delà du challenge intellectuel que cela représente, il peut être intéressant d'obtenir la grande image rapidement sur n'importe quel actif mondial, et choisir sa propre source de données.
Python permet de semi-automatiser cela et peut s'inscrire dans une démarche plus large de promotion du système AT.S (revues multi-actifs, partages réseaux sociaux etc.).
De plus, même si tout outil financier est fortement dépendant des données sur lequel il repose, il est primordial de maîtriser au maximum l'environnement utilisé pour la prise de décision. Vous aurez probablement remarqué certaines différences de chiffres entre ProRealTime et TradingView, ou entres plusieurs sites financiers, sans pouvoir l'expliquer, ce qui peut engendrer de la confusion.

C'est ainsi que le script **PyATS.py** a été développé pour visualiser la grande image du système AT.S.
Voici un exemple avec le CAC40 au 03/10/2022 :



Le rendu est minimal, il est vrai, mais suffisant pour repérer les formes AT.S et la position des prix par rapport aux indicateurs, aux supports et résistances. A la connaissance de l'auteur, le moteur graphique ne permet pas de faire des tracés manuels sur les UTs (Oggy, Jack etc...). Dans ce cas, il faut éditer les images séparément.

NOTES :
1/ Si vous lisez ce document, vous devez savoir un minimum comment installer le système AT.S sur ProRealTime ou TradingView, et comprendre son interprétation.
2/ Les choix technologiques de l'outil dépendent fortement des compétences de l'auteur. Il est certainement possible de mieux faire sur certains aspects (vitesse de calcul, rendus graphiques etc.). L'outil se voulant être Open Source (licence GPL), n'hésitez pas à contacter l'auteur sur le Discord AT.S pour des améliorations, qui seront soumises à validation de l'autorité de régulation du système.

INSTALLATION ET USAGE

1/ Installation

Au moment d'écrire ces lignes (Octobre 2022), PyATS.py fonctionne sous Python v3.10.7.
Il s'appuie principalement sur les librairies externes : pandas, numpy, plotly (et Flask pour l'application Web)

Une fois Python installé, ces librairies peuvent être installées classiquement à partir de l'interface de commandes (« cmd » sous Windows)
python -m pip install pandas numpy plotly

Dans le répertoire du Script, il faut créer les répertoires :

- de stockage des données : *data*
- de stockage des images : *png*
- les fichiers liés a l'application Web : *static, templates*

NOTES: Si vous utilisez un autre OS que Windows, contactez l'auteur pour s'assurer de la compatibilité du script.

2/ Usage

2.1/ Pour un ticker / actif financier

Dans l'interface de commandes (application « cmd » sous windows), il suffit de lancer le script avec la commande :
python pyats.py -t « <ticker yahoofinance> »

A partir de cette commande, le script va télécharger les données YahooFinance, puis générer la grande image dans le navigateur internet par défaut, et sauver l'image dans un fichier PNG pour revue ultérieure.

Et voila !

NOTE : Il faut connaître la structuration des symboles / tickers YahooFinance, par exemple ^FCHI pour le CAC40, ou CAP.PA pour Capgemini etc..

2.2/ Pour un fichier de données financières (ou non)

De plus, il est possible de lancer la grande image directement sur un fichier de données CSV

python pyats.py -f « <nom fichier> »

NOTE : Ce fichier doit contenir les colonnes Date, Open, High, Low, Close. Si vous n'avez que les données en cloture / fin de journées, remplir les quatre colonnes avec la même valeur. En journalier, des petits tirets vont apparaître (comme pour un titre illiquide), mais les chandeliers / bar charts seront bien construits sur les unités supérieures

2.3/ Pour l'application Web

Lancer le serveur de l'application Web avec la commande :
python app.py

Ensuite, aller dans un navigateur et reprendre l'URL mentionnée par le serveur qui devrait être de ce type :
http://127.0.0.1:5000/

Cela charge une page Web minimaliste, où il est possible de rentrer un symbole, comme MC.PA. Une fois le symbole confirmé, cela renvoit vers la page avec la grande image.
http://127.0.0.1:5000/ats

NOTE : il est possible de modifier directement l'URL pour avoir les grandes images sans passer par la page d'accueil, avec :
http://127.0.0.1:5000/ats/<symbole> (ex. MC.PA)

ÉTAPES DE DÉVELOPPEMENT (POUR LES GEEKS)

1/ Téléchargement des données

C'est la première étape. Pour cela il faut d'abord générer l'url de téléchargement de l'historique des prix depuis Yahoo Finance, avec les variables :

- Le symbole / ticker : exemple ^FCHI pour le CAC40
- La période de fin (period2) : la date d'aujourd'hui, convertie au format timestamp
- La période initiale (period1) : 25 ans avant aujourd'hui, converti au format timestamp

Exemple ici avec le CAC :
https://query1.finance.yahoo.com/v7/finance/download/%5EFCHI?period1=873868689&period2=1664860689&interval=1d&events=history&includeAdjustedClose=true

A ce stade, un fichier CSV est sauvegardé localement pour stocker l'historique des données d'un actif et retraiter les données ensuite. Exemple avec le CAC40 :

Date	Open	High	Low	Close	Adj Close	Volume
1997-09-10	2920.100098	2928.100098	2864.399902	2874.600098	2874.600098	0
1997-09-11	2848.199951	2863.5	2829.5	2843.600098	2843.600098	0
1997-09-12	2852.300049	2917.300049	2828	2834.100098	2834.100098	0

NOTES :
1/ il se peut que l'actif ne possède pas autant de données, ou que Yahoo Finance limite l'historique des données au téléchargement. Dans ce cas, les indicateurs sur les UTs longues, en particulier « Année » et « Semestre », seront tronqués.
2/ Si jamais le téléchargement automatique n'a pas fonctionné, il est toujours possible d'aller sur le site YahooFinance et télécharger les données manuellement.
Ou a terme, utiliser d'autres flux de données (en particulier Intraday) pour faire évoluer l'outil.

2/ Génération des indicateurs

Au préalable, le script charge le CSV contenant l'historique des prix, et stocke la table des données dans une structure Pandas, de type *DataFrame*. C'est à partir de cette table que tous les calculs vont se faire.
Les indicateurs techniques du système AT.S sont relativement simples, constitués de :

- Moyenne mobile simple (M7)
- Bandes de Bollinger (M, U, L) dont un des trois éléments n'est autre qu'une moyenne mobile simple de 20 périodes.
- SAR (P)

2.1/ La moyenne mobile, fonction get_sma

Une simple ligne permet de calculer une moyenne mobile sous Python/Pandas, avec la période, et ainsi générer une colonne supplémentaire dans la table des données :
data['Close'].rolling(window=period).mean()

Par contre, il faut générer des colonnes supplémentaires pour distinguer le sens de la moyenne mobile, croissante ou décroissante. Ces 2 colonnes « Up » et « Down » prendrons la valeur de la moyenne mobile, si elle est orientée dans le bon sens, ou 0 quand elle orientée dans le sens inverse.
Ces 2 colonnes permettent de tracer les couleurs vertes et rouges dans le graphe, très utiles dans l'interprétation des supports et résistances.
NOTE : pour comparer la valeur actuelle et la valeur précédente de la moyenne mobile, il est judicieux de passer par une colonne intermédiaire avec un décalage d'une période en arrière.
data['sma'].shift(1)

2.2/ Les bandes de Bollinger, fonction get_bollinger
Les 3 éléments des bandes de Bollinger se construisent dans le même esprit que la moyenne mobile, puisque que la ligne centrale est une moyenne mobile simple de 20 périodes (M), et il sera nécessaire de coloriser le sens de 3 éléments.
Pour le calculs des bandes supérieures (U) et inférieures (L), il est nécessaire d'ajouter des colonnes intermédiaires, avec le calcul du « typicalprice », le point pivot journalier du prix :
(u['Low']+u['High']+u['Close'])/3
Puis la déviation standard (STD) par rapport au TypicalPrice (TP) :
u['TP'].rolling(20).std(ddof=0)
Les valeurs des bandes sont simplement des opérations sur les colonnes M et STD, selon les formules bien connues :

- U = M + 2*STD
- L = M – 2*STD

Comme pour la moyenne mobile, il est nécessaire de créer des colonnes supplémentaires pour déterminer les sens de U et L, et pouvoir les colorier en vert ou rouge sur les graphiques

2.3/ Le SAR AT.S, fonction get_psar
Le code du SAR AT.S est légèrement différent du SAR usuel, puisqu'il est décalé d'une période en arrière. Il faut reprendre le code ProRealTime et TradingView et l'adapter pour Python.

Comme pour les indicateurs précédents, il est nécessaire de distinguer la position du SAR par rapport au prix dans 2 colonnes distinctes, pour colorier les points en vert ou rouge.

3/ Visualisation graphique sur une UT.
A ce stade tous les éléments techniques sont disponibles pour afficher une UT.
La librairie Plotly a été choisie en raison de sa gestion aisée des bar charts financier (OHLC), avec sa fonction native *go.Ohlc*.
Tous les éléments graphiques viennent s'ajouter dans la figure avec les fonctions *add_trace* et *go.Scatter*.
L'ajout des lignes verticales pour Oggy et Jack est faisable via la fonction *add_vline*.

De plus, Plotly permet la gestion d'une matrice de graphiques (*subplots*), ce qui est idéal pour obtenir la grande image du système AT.S

4/ Création des 6 unités de temps
Pour obtenir les 6 unités de temps, il n'est pas nécessaire de télécharger 6 séries de données spécifiques. Pandas permet de regrouper les données Jour grâce à la fonction *resample* , et retrouver les valeurs OHLC correspondantes:

```
    # Other timeframes
    logic = {'Open'  : 'first',
          'High'  : 'max',
          'Low'   : 'min',
          'Close' : 'last',
          'Volume': 'sum'}

    UTs = pd.DataFrame({'Period':['W','1M','Q','6M','Y'],
                'Row':[1,1,2,2,2],
                'Col':[2,3,3,2,1],
                })
    UTs = UTs.set_index('Period')

    for period in UTs.index:
       df = data.copy()
       df = df.resample(period,convention='start').apply(logic)
etc...
```

Une fois les données regroupées dans des nouvelles tables (*DataFrame df ci-dessus*), il suffit d'appliquer le même code que précédemment sur chaque série de données pour obtenir les 6 UTs.

NOTE : sur les UTs longues Semestre et Année, la pertinence graphique dépendra fortement de la présence et de la qualité des données.

5/ Génération de la grande image

La grande image peut être générée a partir d'une figure de 2 lignes et 3 colonnes.

```
fig = make_subplots(rows=2, cols=3,
      subplot_titles=(symbol+' -D', symbol+' -W', symbol+' -M', symbol+' -Y',symbol+' -S',symbol+' -Q'))
```

Les éléments graphiques sont ajoutés pour chaque UT, en fonction de la ligne et colonne cible, selon l'ordre de rotation du système AT.S

Pour l'unité jour, il est nécessaire de cacher les Samedi et Dimanche sur l'axe des abscisses afin de ne pas avoir d'espaces vides dans le graphique

update_xaxes(rangebreaks=[dict(bounds=["sat", "mon"])],rangeslider_visible=False,row=1,col=1)

Enfin, il est possible d'ajouter un titre avec la fonction *update_layout*, puis de visualiser la grande image dans le navigateur internet avec *fig.show()*.

Un fichier png est sauvegardé grâce à la fonction

fig.write_image(img_file,width=1200,height=675)

6/ Opportunités pour le futur

Les possibilités d'évolution du script sont multiples, que ce soit sur le système en lui même (nouvelles UTs pour la petite image ou très grande image, nouvelle source de données etc...), ou sur les fonctionnalités d'automatisation des revues d'actifs.

Dans tous les cas, toute évolution du script sera soumise à validation de l'autorité de régulation du système.